1996-06

The DDD-III: A Tool for Empirical Research in Adaptive Organizations

Kleinman, David L.; Young, Phillip W.; Higgins, Gregory

Center for Advanced Concepts and Technology

Proceedings of the 1996 Command and Control Research and Technology Symposium: Command and Control in the Information Age, p. 827-836
http://hdl.handle.net/10945/64692

Proceedings
of the
1996 Command and Control
Research and Technology
Symposium (1996 : Monterey, Calif)

Command and Control in the Information Age

June 25 - 28, 1996
Naval Postgraduate School
Monterey, California

THE CENTER FOR ADVANCED
CONCEPTS AND TECHNOLOGY

THE DDD-III: A TOOL FOR EMPIRICAL RESEARCH IN ADAPTIVE ORGANIZATIONS

David L. Kleinman*
Phillip W. Young*
LCDR Gregory Higgins**

*Dept. of Electrical and Systems Engineering
The University of Connecticut
Storrs, CT 06269-3157

**Naval Postgraduate School
Code CC, 589 Dyer Road
Monterey, CA 93943

Abstract

Current research involving adaptive architectures for Joint Command and Control (C2) seeks to examine the interactions between task (or mission) structure, and the way in which the organization charged with the mission is itself structured. In order to examine these interactions empirically, a flexible research paradigm is required with which to conduct controlled experiments in a laboratory environment. The 3rd-generation Distributed Dynamic Decisionmaking (DDD-III) paradigm was designed to meet this need by treating an air, sea and ground environment, a variety of task classes, and controllable platforms with subplatforms, sensors and weapons (resources). DDD-III is implemented as a multi-player, real-time simulation running in a UNIX environment.

The design of the DDD-III focuses on the dynamic/execution phase of the mission and allows for manipulation of key structural variables in task and organizational dimensions. The DDD-III has the ability to constrain and/or to manipulate organizational structures such as authority, information, communication, resource ownership, task assignment, etc. This paper describes the new DDD-III paradigm, its extensions beyond the DDD-II, the dimensions of task and organization structure considered, and how they are operationalized.

1 Introduction

Advances in communications technology and computers have made possible real-time distributed Command and Control (C2) on an ever-shortening time scale. Through a Joint forces "Global Awareness," a decisionmaker (DM) can now rapidly access most information available within the organization, monitor actions made by other DMs, and call upon a world-wide database to aid in the decision process. The ways in which DMs within such an organization coordinate their information, resources and activities to attain common goals in a dynamic and uncertain task environment has been the focus of our research over the past 10+ years.

Our approach to studying distributed decisionmaking combines empirical and analytical efforts to develop a model-based program of experimentation with human teams. A first step in this process is to abstract "real world" problems, to bring them into a controlled laboratory environment, wherein we can manipulate selected independent variables. Concomitant with our new efforts, we require a paradigm to capture the salient features and issues of interest for research into adaptive architectures for *Joint* C2. A third-generation, Distributed Dynamic Decisionmaking (DDD-III) paradigm has been developed to fulfill this need. Based upon the earlier DDD-II paradigm [Kleinman and Song, 1990], which was the backbone of extensive empirical research from 1989 - 1995 involving "open-ocean" naval team (distributed) decisionmaking, the DDD-III allows for manipulating many of the variables associated with a Joint battle space.

This paper describes the new DDD-III paradigm, its extensions beyond the DDD-II, and its operationalizing of various dimensions of task and organizational structure. A first application of the DDD-III to simulate a Joint amphibious operation is described, with more detail given in a companion paper [Kemple, Kleinman & Berrigan, 1996].

2 Objectives of the DDD-III

The overall objective of the DDD-III development is to provide a multi-player, real-time environment to support tier-1 (controllable, laboratory based) empirical research for the Adaptive Architectures for Command and Control (A2C2) project. This project, sponsored by the Office of Naval Research, is an initiative to:

- extend Navy decisionmaking research into the Joint C2 arena.
- focus on the relationships between mission structure and organizational structure,
- examine adaptation of organizational architectures in response to changes in mission.

The DDD-III was designed and developed to: create a "realistic" Joint environment within a computer simulation, allow easy manipulation of key structural variables, avoid the need to build/require a technical domain expertise in test subjects and ease scenario design, data collection and retrieval efforts.

3 The DDD-III environment

Several excellent wargamming simulators exist today, such as RESA, JTLS and NTWS, and are used by individual services and Joint commands to train for future conflicts. In general, these tools allow for high-level, theater emulation of tactics in the Joint warfare environment, allowing DMs to determine the changes in outcome that result from specific tactical decisions. These programs, however, have not been designed to allow for manipulation of structure and coordination variables as required to study adaptive organizations. The DDD-III software was developed to abstract the Joint level decisionmaking process found in wargamming simulators while allowing for significant data collection. While not a true tactical model, the DDD-III is conceptualized as a bridge between the real-world Joint (operational) environment and the laboratory.

3.1 Joint warfare framework

The Joint warfare concept was used to frame the DDD-III paradigm, with a Joint Task Force (JTF) commander and subordinate DMs, each responsible for different but overlapping elements in a common battle space. Our first application involved six subject DMs — a leader (the CJTF) and the primary decisionmaker for each of five subordinate units or commands (GCC/MCC, CVBG, ARG, and two MEU-SOCs) — in an environment that simulated a Joint (Navy-Marine) amphibious operation.

The DDD scenario requires subjects to follow a set of predefined mission requirements, while simultaneously defending one or more "penetration zones," and possibly their own assets, against potential land, sea, and air threats. The scenario in the DDD-III is typically crafted by the experiment designer to uncover key issues in the organization's decisionmaking and/or coordination processes. For example, limiting or constraining the assets available to the organization will cause DMs to compete for assets in order to accomplish their *local* objectives — especially if the scenario has been designed so that several DMs have simultaneous need for a resource. Competition over assets is a major issue in the real-world Joint environment, where priorities must be adjusted dynamically to allocate limited organic and non-organic resources in such a manner as to achieve the *overall* mission objective.

3.2 Typical mission structure

The various complex mission requirements faced by the Joint force are represented in the DDD-III as a "task" structure. Here, tasks are defined as abstracted activities (or things to do) and can range from taking a beach, clearing a minefield, or identifying and (if necessary) prosecuting a hostile contact. The scenario designer can create a template of tasks, with defined attributes, threat capabilities and movements, that are linked together by time, space and/or precedence to provide a theater level mission for the organization to accomplish.

The DDD gives each DM responsibility for a specific mission area (set of tasks), as defined by the Joint Operation Order (OPORDER). The DMs must maneuver their assets/platforms so that tasks are brought within sensor or weapons range. As the scenario unfolds, a DM who is prosecuting tasks with his own platform(s) might require additional assets from other units within the JTF. Thus, it becomes important for the team to coordinate the allocation of assets and activities — both for information gathering and task prosecution — in order to ensure success of the overall mission.

An overriding hypothesis to the A2C2 research

is that changes in the task structure will drive changes to the organizational structure. The DDD-III has the ability to explicitly operationalize many of the relevant organizational dimensions within a JTF. By allowing manipulation of the task structure, task assignment and responsibility, the command structure and communications structure of the decisionmaking team, etc., the DDD-III becomes a powerful empirical instrument.

4 Task dimensions

During the real-time playout of an experimental run, tasks appear, move/maneuver, and disappear according to a user-scripted scenario. The user has the ability to define various dimensions of task structure in order to closely align the DDD scenario with the prescribed "mission" for the JTF. The characteristics of each task — its class, attributes, and resources required — can be tailored to specify the threat (such as a fighter, minefield, etc.), and to create intra-team conflicts for assets needed in the prosecution of those tasks.

4.1 Task type and class

Tasks are objects, categorized as one of three *types*: air, surface, or ground. Within each type, tasks are further divided into unique *classes* as per the scenario requirements. This allows the experiment designer to create, within a given task type (e.g., ground), a variety of classes such as: clear minefields, armored columns, take high-ground, surface-to-surface/SCUD missile sites, and even "false" tasks. Each (class of) task/activity can be assigned an (initial) priority, along with the DM who has responsibility for that task, and which DMs within the team have initial authority (or ability) to prosecute the task(s).

4.2 Task attributes

Associated with each task is a set of (numerical) attributes, written as a vector $A = [a_1, a_2, \ldots, a_n]$, that defines the various characteristics of the task quantitatively. For example, the attributes could include speed, (ground, sea, or air) weapons potential, evasive ability, IFF status, etc., depending on the specifics of the problem being studied. The DDD-III assumes that the first two attributes of any task are its "value" and processing time requirement, respectively. The true attribute vector for all tasks within a given class is drawn initially from a Gaussian distribution, with user-specified mean and standard deviation. The software allows the designer to tailor the individual values of these attributes on a task-by-task basis, if so desired.

Sensors on board the various platforms obtain measurements of task attributes, provided the tasks are within their sensor range. The DDD-III has the capability to apply any level of "noise" and/or "bias" to these attribute measurements, so that the true values of the attributes are obscured from the DMs. If the attributes are needed to identify task class (e.g., to discriminate a hostile from a neutral, or to identify the class of an incoming patrol boat), the lack of "perfect" measurements will result in mis-identifications. The means and standard deviations of the sensor noises, and values of the (circular) sensor ranges can be functions of both task class and platform class.

4.3 Task attributes-to-resource mapping

The attribute vector gives requisite information regarding a task's characteristics. With each task there is also associated a resource vector $R = [r_1, r_2, \ldots, r_m]$ that defines the resources required to successfully process (attack) that task. The user-defined elements of R can be viewed as generic weapons' requirements such as ground suppression, mine clearing, anti-air potential, etc., depending on the level of the problem being simulated. Default values for the r_i are generated within the DDD-III via a (user-defined) mapping function $\mathbf{f}(A, ID)$ that generates resource requirements from the attributes of the task and its class. The DDD-III software also allows the scenario designer to overide these values, and give a specific resource requirement to any individual task.

In the play of the game the subjects are automatically given the results of the attribute-to-resource mapping based on their current estimates of A and class ID. Imprecise knowledge of a task's attribute values and/or a mis-identification of class ID will therefore result in a DM's obtaining incorrect estimates of that task's resources required.

Attributes and resources are the DDD's lowest-tier constructs for representating information/data and weapons systems, respectively. We believe that these are valid constructs for hierarchical aggregation of activities and assets. [An experimenter should not have to build scenarios from the radars (or "bottom") up to study problems at a large force/component level.] By treating task attributes and task resources as two separate vectors, the DDD-III can be used to study pure (distributed) information processing problems, pure (distributed) resource allocation problems, or a hybrid. For example, in a pure resource allocation context the

attributes can be associated 1-to-1 with the resources ($r_i = a_i$), and the "measurements" of the a_i can be made noise-free. Our first experiment with the DDD-III used just such a construct.

4.4 Task precedences/prerequisites

A crucial element in scenario design is the ability to assign prerequisites (or corequisites) for the accomplishment of tasks. This is an important dimension of task structure as it defines correlations and coordination requirements among the individual activities that combine to comprise the mission as a whole. The earlier DDD-II considered each task as an independent threat (a "mosquito"), with no consideration given to tasks that are not threats per se. (An attempt to develop multi-operation tasks within the DDD-II had some partial success.) The precedence structure implemented in the DDD-III allows for task fan-outs and fan-ins. In the former the lack of, or delay in, accomplishment of a (single) task can inhibit action on many tasks, e.g., taking a high ground can be a prerequiste for follow-on introduction of forces. In the latter case a number of (possibly diverse) activities may need to be accomplished before, say, a final attack on an airfield can be mounted.

A "real" task environment does not usually inhibit a DM from taking an action. In such cases, if actions are done out of sequence, not according to plan, the consequences may be severe. In our research, where task structure is to be among our independent variables, we need the ability to control task structure to a sufficient degree as to prevent teams from creating totally arbitrary paths to achieve a final objective. The specification of a prerequisite mission/task structure (e.g., clearing minefields before landing on a beach), along with the DDD-III's forcing adherence to this structure, provides this needed ability.

Parallel processing of a task by several DMs is now supported in DDD-III. This is a key element in scenario design when assets that are owned by different organizational units are to be coordinated in time/space to mount a simultaneous attack. An assault on a hill by a Marine unit, supported by CAS, artillery, and Naval gunfire is an example of such a multi-resource activity. The DDD-III "scores" the effectiveness of the attack as a function of the sychronicity (and correctness) of the allocated assets.

4.5 Task spawning

Missions in the real-world often develop secondary mission objectives during prosecution. The DDD-III provides the capability to "spawn" secondary tasks from primary tasks, allowing the scenario designer to continually provide dilemnas and conflicts for the organization's decisionmakers. This feature of the DDD-III also prevents the team from being led down a single path in the scenario, solely for the sake of the hypothesis under test.

Task arrivals and disappearances in the DDD-II were only time-driven, and not event-driven. Thus, it was not possible to script enemy actions in (partial) response to the actions of the subject team. During the course of designing the first experiment it was found necessary to include enemy counterattacks, medical evacuations, and other "tasks" that would be "spawned" by a specific action such as landing on a beach. The DDD-III implements an extended task structure by allowing a primary task to "spawn" a secondary task either upon attack of the primary task, or upon disappearance of the primary task. As an example of the latter, the failure to identify and prosecute an enemy submarine by a specified time would result in a cruise missle attack upon the CVBG.

5 Platforms and subplatforms

A platform (or asset) is a basic element of the DDD paradigm, that carries sensors and resources (weapons). Examples of platforms are ships, helicopters, ground units, bases, etc. A platform may also carry subplatforms. For example, a carrier can contain helicopters and various fixed-wing aircraft, and the helicopters can carry sonobuoys, etc. In this way a platform/asset structure can be nested down to any desired level of detail. An amphibious landing can be modeled by a platform-subplatform structure using sea-going platforms that carry ground units for launch at a shoreline. These ground units may then have sub-units of their own.

The team identifies and prosecutes tasks by allocating/scheduling its platforms, ideally to make best use of their generic sensors and resources. A platform can be controlled (i.e., geographically repositioned, commanded to attack, etc.), only by the current owner of that platform. Assets have sensor and weapons' ranges, which generally depend on task type and platform class. A new feature in the DDD-III is that each platform now has a "be-attacked" range, within which specified task classes can inflict damage to it. The values for all ranges can be manipulated via the paradigm depending on the experimental requirements.

5.1 Platform type/class definition

As done for tasks, platforms are categorized first by *type*: air, sea or ground, and then by *class*. The number of platform classes and their individual characteristics depend on the requirements of the experiment. All platforms of a given class will have the same parametric features with respect to sensors, weapons, maximal velocity, subplatforms they carry (class and number), etc. The only difference among platforms within a given class is their initial owner and location.

5.2 Subplatforms

Via subplatforms, located on board parent platforms, the DDD-III more accurately models the assets available in the Joint world today, and allows a hierarchical asset structure. A subplatform does not become an independent platform until it is "launched" from the parent platform. The DM who owns the parent can launch one or more subplatforms that will become available after a specified launch time delay. Ownership of the subplatform "child" can be specified independently. For example, F16s on board a CV (which is owned by the CVBG), could be owned by the JFACC when they are launched.

A subplatform is only effective (available for use) for a limited time period. Subplatforms are either returnable to their parent (such as helicopters), or non-returnable (such as sonobuoys). Once returned, a subplatform is not available again until a recycle time has elapsed. Subplatforms can also be designated as reusable or non-reusable. An asset that is non-reusable can only be used to attack once, whereas a reusable subplatform can attack multiple times, within its availability window. All parameters relative to subplatform structure and availability are user-specified in the DDD-III.

5.3 Platform sensors

Each platform has three types of generic sensors for obtaining information on air, surface, and ground tasks, respectively. These sensors have specified effectiveness ranges (modeled as circular regions) for task detection, measurement, and identification/classification which depend on platform class. However, the DDD-III gives the user the ability to fine-tune these ranges according to individual *task* classes. Thus, it is possible to implement an engineer platoon that "sees" mines at a further range than does a tank column. For a platform to "see" a task, it must be within the platform's detection zone; to obtain a measurement of task attributes, the task must be within the measurement zone; to obtain task class identification, the task must be within the classification zone. As noted earlier, measurements of task attributes can be "contaminated" by noise.

5.4 Platform resources

Platforms also contain weapons (or general resource capabilities) for processing tasks. The resources on an individual platform (or subplatform) are defined by a generalized resource vector, or vector of generalized combat capabilities, $R = [r_1, r_2, \ldots, r_m]$, where the elements r_i are the same as those associated with task processing requirements. A platform can be used to attack any task, but the range of the platform's weapons depends on the task type (or class). In order to engage a task, a DM must move one or more selected (sub)platforms within range of the task for (identification and) attack. Once an attack starts the platform(s) are tied-up for a length of time equal to the task's processing time required, a_2.

The resources on the platforms assigned to attack a task must meet or exceed those required by the task for the attack to be rated as successful. If the summation of allocated resources, on an element-by-element basis, is less than the required amount, the attack will achieve partial success at best. By giving a specific task a value of R the DDD-III establishes what (mix of) assets with their corresponding Rs suffice to correctly process that task. In this vein, platforms can also be made job-specific. For example, mine-clearing helos could have values for, say, r_5 corresponding to those required for mine clearing tasks, but other assets would have lower values of r_5, or zero.

The expression for determining the accuracy of the attack on task j, P(j), can be user-modified. After each attack the DDD-III gives a "gain" to the team of $V(j)*P(j)$ and a "loss" to the team of $V(j)*[1 - P(j)]$, where $V(j) = a_1$ = task value. As noted earlier, the resources required for task processing are a function of task attributes. The DDD provides each DM with an estimate of resources required, using his/her current estimate of task attributes and class ID. Thus, accurate task information processing is a precursor to correct resource allocation. For example, attacking a task defined as a non-threat will always give $P(j) = 0.0$.

There is no attrition in the current implementation of the DDD. A poorly done attack, an enemy penetration of a defense zone, etc., result only in point loss, and not in a loss of assets or asset

capability. This was done so that the resources available are not a (uncontrollable) function of the team's sample path through the scenario. This is critical to controlled experimentation since the task/mission structure is generally predicated on asset availability. If task structure is to be an independent variable, the total assets available should remain constant — or else the team should be allowed to restructure the mission.

6 Organizational dimensions

The organizational dimensions include authority (or command) structure, resource structure, information structure, communication structure, expertise structure, goal structure, etc. The *overall* organizational structure must maintain a general congruence with the task structure to assure that DMs assigned to selected (sub)tasks have the resources and information required (or can obtain them via a request chain), can communicate with other DMs with whom they need to coordinate, and that the "chain of command" that is established facilitates successful mission completion. Thus, organizational design is a multi-dimensional problem, where seldom can one dimension be changed without considering concomitant changes in other (supporting) dimensions.

The DDD-III has been designed to facilitate changes to the key organizational dimensions via user input parameters. Thus, future research that will allow subjects to manipulate organization structure as a *dependent* variable during the course of an experimental trial should be accommodated with relative ease.

6.1 Command/authority structure

Military organizations adopt a command hierarchy to structure the overall decisionmaking process. The DDD-III allows for tailoring a "chain of command," provided that each DM in the organization has no more than one "boss". We use a general acyclic tree structure to code the command hierarchy by giving to each DMj a single integer $c(j) \leq j$ that defines to whom DMj reports. Command structures ranging from totally disconnected [c(j) = j], to totally flat [c(j) = 0], to totally serial [c(j) = j-1] can be treated in this construct. For example, in our experiment wherein two units reported directly to the CJTF (DM0), and two other units reported to a common mid-level functional commander (DM1), we had C = [0,0,0,0,1,1].

This generalized authority or command structure is operationalized via task assignment and platform/asset assignment capability. As only those DMs assigned to a task are allowed to prosecute it, a leader can control dynamically the coordinated actions within his (sub)team, and reassign platforms according to mission need.

6.1.1 Task assignment

In the DDD-III a DM can (re)assign or co-assign a specific task to any subordinate DM in his/her sub-organization (subtree). Assignments cannot be made upward nor laterally. Assignments are constrained in that: i) a task cannot be taken on unilaterally, i.e., the task must have already been assigned to someone in the subtree, and ii) the task cannot be given away unilaterally, i.e., assigned away from all DMs in the subtree.

6.1.2 Platform assignment

The DDD-III operationalizes a DM's authority over the assets in his/her sub-organization in two ways. He can *advise* a subordinate DM to transfer his asset(s) to another DM, or he can independently and unilaterally *force* a transfer action. If an asset is owned by someone outside of his sub-organization, then all that a DM can do is to request use of that asset (from the owner or via the chain of command).

6.2 Resource structure

This defines both platform ownership and the "rules" by which platforms can be requested, transferred, and accessed. At any given time each platform is controlled by the DM who owns it. If a platform is defined as transferrable, it can be transferred during the simulation from one DM to another — either by the owner or by any of the owner's superiors acting through the chain of command. Other platforms can be defined as non-transferable, e.g., the amphibious ships (which belonged to the ARG), and the aircraft carrier (which belonged to the CVBG) in experiment 1. The transfer of an asset from one DM to another requires a finite (user-specified) time delay, during which period the asset is "locked up," or inhibited from accepting other commands.

A subplatform nesting structure, wherein the "owner" of a platform is not necessarily the owner of its subplatforms, is supported within the DDD-III. Some subplatforms can also be assigned permanently to the parent asset irrespective of ownership, such as a ship's self-defense assets.

6.3 Information structure

The information (access) structure describes how

data collected by platform sensors is distributed to the various DMs in the organization. The information structure is operationalized in the DDD-III by defining an information network. Each DM can be assigned a level of "tie in" to the information network depending on task type. Each task class can be tailored to be viewed differently by each platform class, allowing for a truly robust and diverse information architecture.

For example, a leader (who is is responsible for global coordination) can be provided with global information, while other DMs can be given more detailed local information depending on their local decisionmaking responsibilities. A centralized, partially centralized or decentralized information structure can be created by setting the different network "tie in" levels by task/platform class for different DMs. In the first experiment a common operation picture (COP), also known as the "Global battlespace awareness" concept, was used. Each DM was able to see in real-time the location of all contacts detected in the task force operations area. However, detailed attribute data needed for resource allocation and/or classification was sometimes only provided to the local DM assigned to that task.

6.4 Communications structure

The communication structure specifies who can send messages to whom, and also includes parametric data such as receipt delays, equipment delay and "bandwidth". By preventing communications between certain units and forcing communications up and down a specific chain of command we are able to observe the impact of different communications structures not just in isolation, but more interestingly in its relationship (interaction) with other structural dimensions.

In running the DDD software we generally inhibit verbal exchanges among DMs in order to ease subsequent analysis of the communications data. Thus, computer-mediated communications is the only way by which DMs can share local information about task attributes and ID, request assets, and coordinate actions. The communication among DMs is effected through the graphics user interface via a set of preformatted commands:

- request information: ask a DM to send his local information about a task.
- transfer information: transfer one's local information about a task to another DM.
- request platform: ask a DM to transfer the ownership of a platform to another DM.
- transfer platform: inform another DM that a platform is being transferred to him (or that an earlier request is being denied).
- coordinate action: several choices are available in this command and can be easily changed to accommodate specifics of the simulation:
  - ask another DM to either handle, support, or ignore a task.
  - tell other DMs of one's intent to handle, support, or ignore a certain task.

Copies of all messages sent by a DM (except for information requests/transfers) are automatically sent to that DM's superior in the chain of command. This keeps the higher-level DM apprised of the needs/activities of his subordinates, and is an attempt to capture overheard (open) message flow on a DM command net.

To represent communication and data processing delays, the DDD places a time delay on a message transfer. To model a limitation on channel capacity (or channel access), the number of communications (N) in a fixed time window (T) can be specified.

6.5 Other organizational structures

Other organizational structures, not fully implemented in the DDD-III include expertise and goal. The expertise structure refers to the level of capability of various DMs to process information, assess the situation, and/or process tasks (or use assets to their full effectiveness). The DDD can give DMs differential capabilities in task identification and attack, but this is a very myopic view of the broader issue of "expertise".

Goal structure per se is not implemented in the DDD, although the accuracy of task processing can be aggregated on a DM-by-DM basis, or by responsibility. Feedback (and reward) to subjects can thus be tailored to something other than a single, common, team performance measure or reward structure.

7 Measures collected

Using the DDD-III software a designer can manipulate a variety of independent variables for the study and evaluation of different task and organizational structures. The number and type of *dependent* variables this paradigm can handle is quite flexible, and basically fall into one of two (not necessarily independent) categories: performance measures and process measures.

Performance measures deal with team score and

mission effectiveness. These include: team timeliness measures such as latency, slack time, etc.; team performance quality measures such as accuracy and efficiency in either information processing or resource allocation. Process measures describe the mechanisms by which the team attained its performance. Among these, decision strategy measures are often categorized by individual DM, by task type or class, by timeline according to when actions were taken, etc. Thus, these measures include distribution of load among the individual DMs, amount and pattern of resource transfers, resource utilizations, reneging (not performing certain tasks), etc. Coordination strategy measures describe the means by which the team strategy was effected, i.e., how the DMs dynamically supported the strategy that they utilized. These measures largely involve communications usage (e.g., resource and/or information request), and communication patterns among DMs.

7.1 Data collection and retrieval

The DDD-III produces two data files at the conclusion of a run: a dep_file and a log_file. The dep_file is an aggregation of actions and/or results on individual tasks by DM, by class, etc., as well as communications averaged by pattern and type. The dependent variables recorded in this file are generally user-specified vis-a-vis the current context. For example, in the first experiment these variables included: 1) the number of attacks, and their average accuracy and latency, by task class and DM, 2) number of penetrations by task class and into which zone, 3) obstacles (e.g., minefields, SAMs) that were not sucessfully avoided or cleared, etc. Clearly, these are "micro-measures," collected and averaged by the DDD-III on a task-by-task basis. These measures can, in theory, be aggregated into higher level or global measures that deal with overall task processing and asset allocation. Access to the DDD source code provides this ability, provided the researcher can describe analytically the desired measures.

All essential operations taken by the DMs are recorded in the log_file. This file can be used to generate dependent variables and statistics, not already included in the dep_files in a post hoc manner. This file can also be used to automatically replay the game for review/demonstration, for training, or to compute other dependent variables not originally collected in the dep_file when the experiment was first run.

8 Implementation

The DDD-III runs over a network of UNIX workstations. The software has roughly 60,000 lines of C code; graphics routines are written in MOTIF. Implementation consists of four major parts: Global, Local, User-interface, and Scenario Generator. *Global* is the heart of the software — it is a run time communication/control and data processing center. *Local* is in charge of object (platforms, tasks) control and command processing. Through the (graphical) *user-interface* the subjects interact with the run-time software through a combination of pull-down menus and pop-up windows. All subject commands and entries are mouse-driven, with no real-time keyboard entry of text/numbers required.

The paradigm runs concurrently on as many workstations as needed (up to 30 DMs are currently supported), with all control and communication traffic carried over Ethernet using XDR protocol. This is a central controlled distributed system. The central controller, Global, takes care of synchronizing all the Locals and maintaining data consistency among Locals' (distributed) databases.

8.1 Scenario generator

The scenario generator is the tool through which the user translates the Joint mission requirements into DDD-specific constraints, and defines the Joint warfare "game" world — from the ideal (or "global awareness") situation to the more realistic situation in which a hostile adversary introduces uncertainties and deceptions.

The scenario generator requires four major categories of inputs:

1) general information
2) resource and task information
3) state information
4) maneuver information

General information includes such features as the location of land area, numbers of DMs, command hierarchy, communications connectivity, game time, number and location of penetration zones to be defended, etc. In resource and task information we describe the general characteristics of both platform and task *classes*, such as speed, weapons/sensors, attributes, resources, the DMs who can process the task, etc. In state information we give *specific* resources and capabilities to the platforms (including their initial location and owner), and *specific* attributes to the tasks, to override class default values set earlier. Here, we also define any

task precedence requirements, and task spawning conditions. In maneuver information we describe the task arrival times, initial positions, velocities, and their maneuvers (which are assumed to consist of straight line constant velocity segments).

TASK ATTRIBUTE VALUES

TASK	VALUE	TIME	AIR	SEA	GRND	HOLD	MINE	ARMOR	MED	ENEMY
0 HILLS	10.0	20.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	-
1 AIRPRT	30.0	20.0	0.0	0.0	10.0	10.0	0.0	0.0	0.0	-
2 SEAPRT	30.0	20.0	0.0	0.0	10.0	10.0	0.0	0.0	0.0	-
. .	.	.	.	.	.	.	.	.	.	.
. .	.	.	.	.	.	.	.	.	.	.
20 MEDVC	5.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	-
21 SWAMP	2.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
22 SILK	15.0	20.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	*

*The value of this attribute can be set to either 0: positively identifiable as a decoy, or 1: positively identifiable as an enemy.

PLATFORM RESOURCE VALUES

PLATFORM	AIR	SEA	GRND	HOLD	MINE	ARMOR	MED
1 Carrier based fighter aircraft	5	2	0	0	0	0	0
2 Land based fighter aircraft	5	1	1	0	0	0	0
3 Carrier based attack aircraft	0	5	5	0	0	5	0
. .	.	.	.	.	.	.	.
. .	.	.	.	.	.	.	.
23 Landing craft †	0	0	0	0	0	0	0
24 Armored vehicles	0	0	5	5	0	1	0
25 Ground bases (in region)†	0	0	0	0	0	0	0

† These assets contained only subplatforms that could be "launched" by the various commanders

FIGURE 1. EXAMPLE VALUES FOR PHASE ONE DDD-III SIMULATION

8.2 Example

The implementation of the experiment described in [Kemple, Kleinman & Berrigan, 1996], and alluded to throughout this paper, used 24 task classes and 12 platform classes. The scenario involved a Joint amphibious landing, which required the use of a carrier battle group (CVBG) and an amphibious readiness group (ARG) along with their various subplatforms, plus non-organic theater-based assets. The mission (task structure) required the simultaneous taking of two beachheads, transiting through enemy territory, and a final coordinated attack on both an airport and a port.

The task attribute vector was 10-dimensional; the platform resource vector was 7-dimensional. Figure 1 gives a representation of these vectors, with associated values for several task and platform classes. Element a_{10} in the attribute vector was the "enemy" attribute, which was only measurable by one platform — the long range SR71 reconnaissance aircraft. By carefully tailoring the attribute vectors of individual tasks we created the possibility of false targets, such as reported Silkworm missile sites, that were only confirmable by the use of recon aircraft. This tailoring of attributes-to-assets was just one of the ways that competition (among DMs) for non-organic assets was produced.

Figure 2 is a snapshot of a typical DM's screen part-way through a scenario. Subjects can zoom-in or out as needed. The land area is the shaded rectangle in the left-hand portion. Roads can be seen connecting the two beachheads with both the airport and the port objectives, all four of which have a "penetration" or defense zone clearly visible. The land areas not connected by roads are covered with swamps, effectively preventing off-road travel as per the scenario constraints. In the ocean areas there is the CVBG to the north (not seen here) and the ARG to the south, both with their associated "penetration" or defense zones.

Some of the task objects visible include an enemy tank company on the northern roadway, a Frog launcher near coordinate (3.5, 60), an unknown air contact A?-271 at (37, 84), an unknown sea contact at (36, 37), etc. The JTF's assets are shown as squares (platforms) or circles (subplatforms) with identifying labels, and are color-coded (by DM to

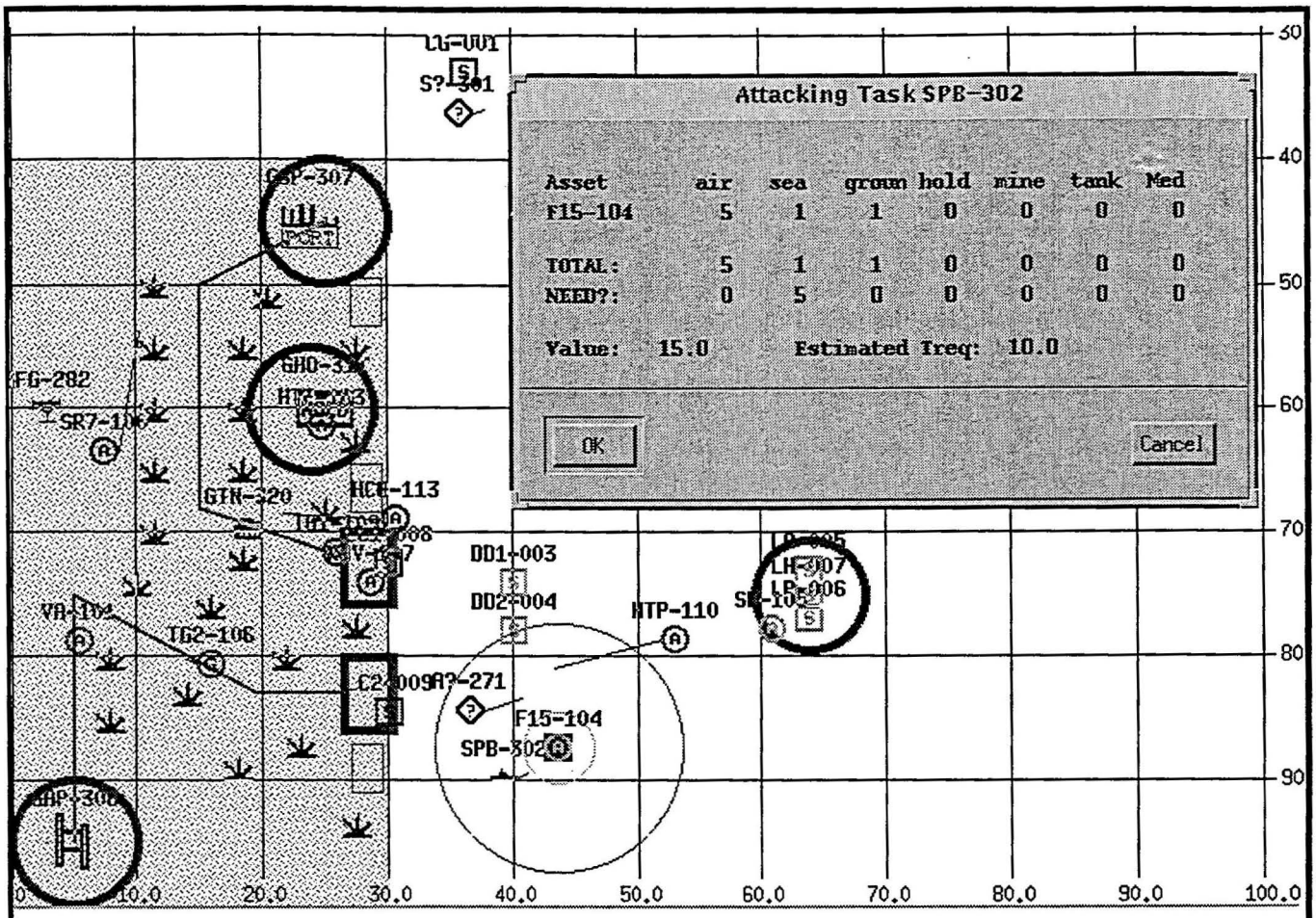

FIGURE 2. SNAPSHOT OF A DECISIONMAKER'S DISPLAY

indicate owner). At the time of the snapshot this DM was about to attack an identified enemy patrol boat SPB-302 that was within range of his F15-104. This is not a well-matched attack: The task requires 5 units of sea combat power, but the F15 has only 1.

Through clicking on objects/icons, the DM accesses various pull-down menus and pop-up windows to control assets, launch subplatforms, obtain task and asset status information, and communicate with other DMs.

9 Summary

We have developed a generalizable DDD-III paradigm rooted in Joint C2. It characterizes team decision processes in which limited, shareable assets must be allocated to identify and process tasks in a dynamic and uncertain environment. It is a research tool amenable to systematic and scientific study of structural adaptation within a hierarchical organization, while emulating many of the essential features of Joint operational decisionmaking.

The first application of the DDD-III was to study a Joint (Navy-Marine) abstracted JTF amphibious landing operation with both maritime and ground components. This "pilot" experiment was conducted at the Naval Postgraduate School using 6-person teams in early March 1996.

References

[Kleinman and Song, 1990] David L. Kleinman and Anlan Song. A Research Paradigm for Studying Team Decisionmaking and Coordination in *Proc. 1990 JDL Symposium on Command and Control Research*, Monterey, CA, June 1990, pp. 129-135.

[Kemple, Kleinman & Berrigan, 1996]. William G. Kemple, David L. Kleinman and Michael C. Berigan. A2C2 Initial Experiment: Adaptation of the Joint Scenario and Formalization in *Proc. 1996 Command and Control Research and Technology* Symposium, Monterey, CA, June 1996.